8-14-00 A

08/10/00 JC907 U.S. PTO

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (4/98)
Approved for use through 09/30/2000 OMB 0651-0032
Patent and Trademark Office. U S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	RCC-001
First Inventor or Application Identifier	Sujith Arramreddy
Title	"Peripheral Component Interconnect Arbiter..."
Express Mail Label No.	EL597198174US

JC864 U.S. PTO 09/637844 08/10/00

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. [ ] * Fee Transmittal Form (e.g., PTO/SB/17) (Submit an original and a duplicate for fee processing)
2. [X] Specification [Total Pages 16] (preferred arrangement set forth below)
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings (if filed)
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [X] Drawing(s) (35 U.S.C. 113) [Total Sheets 5]
4. Oath or Declaration [Total Pages]
   a. [ ] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d)) (for continuation/divisional with Box 16 completed)
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

* NOTE FOR ITEMS 1 & 13 IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. [ ] Microfiche Computer Program (Appendix)
6. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [ ] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement (when there is an assignee) [ ] Power of Attorney
9. [ ] English Translation Document (if applicable)
10. [X] Information Disclosure Statement (IDS)/PTO-1449 [X] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
13. [ ] * Small Entity Statement(s) (PTO/SB/09-12) [ ] Statement filed in prior application Status still proper and desired
14. [ ] Certified Copy of Priority Document(s) (if foreign priority is claimed)
15. [ ] Other: ....

16. **If a CONTINUING APPLICATION,** check appropriate box, and supply the requisite information below and in a preliminary amendment:
[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: ______/______
Prior application information: Examiner ______ Group / Art Unit: ______
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[X] Customer Number or Bar Code Label: 022888 (Insert Customer No. or Attach bar code label here) or [ ] Correspondence address below

Name	BEVER, HOFFMAN & HARMS, LLP				
Address	2099 Gateway Place, Suite 320				
City	San Jose	State	CA	Zip Code	95110-1017
Country		Telephone	(408) 451-5907	Fax	(408) 451-5908

Name (Print/Type)	Jeanette S. Harms	Registration No. (Attorney/Agent)	35,537
Signature		Date	8-10-00

Burden Hour Statement: This form is estimated to take 0 2 hours to complete Time will vary depending upon the needs of the individual case Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

PERIPHERAL COMPONENT INTERCONNECT ARBITER IMPLEMENTATION WITH DYNAMIC PRIORITY SCHEME

Sujith Arramreddy
Ragu Appanagari

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a peripheral component interconnect (PCI) arbiter, and in particular to a PCI arbiter with a dynamic priority scheme.

Discussion of the Related Art

A peripheral component interconnect (PCI) bus is an industry standardized expansion bus that conveys much of the information and signals of a computer system. Optimally, when the computer system executes its programming, information should flow as fast as possible to ensure the computer is responsive to the user. To prevent mistakes in the transmission of that information, a PCI bus design includes a special logic circuit and associated signals to control the flow of that information.

Specifically, a typical PCI bus allows a bus controller, also called an arbiter, to control bus transfers. A device that takes control of the bus to handle its own transfer is termed a "master", whereas a device that receives data from the master is termed a "target". The arbiter uses an algorithm to determine which master can take control of the bus and the time period of that control.

Arbitration must resolve the competing goals of fairness and priority. Fairness requires that one master

should not be allowed to monopolize the bus. However, priority requires that, under certain circumstances, predetermined masters should use the bus more often to accomplish time critical goals. Some typical algorithms used by arbiters are the Single-Level Round Robin, the Multi-Level Round Robin, the Least Recently Used, and the Priority Based approaches.

In the Single Level Round Robin approach, a small unit of time, i.e. a quantum, is defined. All processes (associated with specific masters) are put in a circular queue. The arbiter follows the queue, and allocates the master's use of the bus to accomplish the process for a time interval of one quantum. Any new process is added after the last process in the queue.

If the process finishes before the end of the quantum, the master releases the bus voluntarily. However, if the process is still running at the end of the quantum, the master is preempted and the process is added to the end of the queue. In either case, the arbiter assigns the bus to the next process in the queue.

In the Multi-Level Round Robin approach, at least two circular queues are formed. For example, assuming first and second queues are formed, processes that use the bus frequently are placed in the first queue and processes that use the bus less frequently are placed in the second queue. Processes in the second queue have equal access to the bus, if in the same queue. However, the processes in the second queue, as a group, have equal access to the bus as each process in the first queue. In other words, the processes of the second queue effectively form a "super process", wherein the super process is deemed to be one of the processes in the first queue. Thus, for every round of the

first queue processes, one process of the second queue is performed. In this approach, if the process finishes before the end of the quantum, the master releases the bus voluntarily. However, if the process is still running at the end of the quantum, the master is preempted and the process is added to the end of the appropriate queue.

In the Least Recently Used approach, an arbitrary queue is formed. The arbiter initially follows the queue and allows each process to finish before allowing the next master in the queue to get control of the bus. However, if the arbiter receives a request for bus control from a master not next in the queue, the arbiter gives control of the bus (after the completion of the process running) to the master that has least recently used the bus.

Finally, in a priority-based approach, the arbiter determines bus control based solely on the priority of the associated process performed by the master. In this approach, each process completes before the next process is initiated.

Each of the above-described approaches has its disadvantages. For example, in both the Single- and Multi-Level Round Robin approaches, a quantum may not allow a master time to finish a critical process. Therefore, completion of that critical process may take several complete cycles of the queue, thereby introducing significant inefficiency in the system.

In the Least Recently Used approach, processes that are non-critical get more bus control than in other approaches. Although this allows less frequently used processes an opportunity to complete, it also necessitates losing time resources for other more critical processes.

Therefore, this approach also frequently results in significant inefficiencies in the system.

In the Priority Based approach, depending on the task to be completed by the system, non-critical processes may only rarely be run. Although these non-critical processes may relate only to processes such as expansion bus requests, sporadic or even non-completion of these processes may contribute to some system inefficiency.

All of these approaches use static algorithms to determine control of the bus. As noted above, each of these static algorithms fails to provide the flexibility to optimize system efficiency. Therefore, a need arises for a flexible, arbitration scheme that optimizes system efficiency.

SUMMARY OF THE INVENTION

The present invention provides a dynamic priority scheme that uses information including the status of the target and data availability in deciding which master should be assigned ownership of a PCI bus, thereby optimizing performance and utilization of the PCI bus. Specifically, the present invention provides multiple levels of master priority. In one embodiment, three levels of priority are provided: HIGH, MEDIUM, and LOW.

Once a request from a master is posted, an arbiter in the system issues a provisional grant signal to the master. At this point, the arbiter in the system assigns the requesting master a MEDIUM priority and forwards the request to the target. The arbiter then determines if data is available from the target. If data is available, then the arbiter reassigns the requesting master a HIGH priority. However, if data is not available, then the

arbiter reassigns the requesting master a LOW priority and ignores the requesting master until the arbiter is notified that data is available from the target.

In accordance with the present invention, each target includes a memory interface to facilitate the prioritization process. Specifically, upon receipt of a memory access request from a master (via the arbiter), the target stores this request in a request queue, which forms part of the memory interface, and then proceeds to capture the information needed to complete the access of the memory. After the data is copied in the request queue (i.e., the data is now available), the target generates a master ID for triggering a status change of the requesting master. In a preferred embodiment, the target generates the master ID using the request from the master (called a modified request). This master ID is then provided to the arbiter.

After the arbiter receives the modified request, the arbiter changes the priority of the master to HIGH and, assuming the PCI bus is available and no other masters have an earlier high priority, sends the requesting master a final grant signal, thereby allowing the master to take control of the PCI bus. Note that if the PCI bus is currently being controlled by another master or if other masters have an earlier high priority, then the arbiter sends the final grant signal at the next earliest time period after the process performed by the last controlling master is complete.

To further increase the efficiency of the present invention, the request queue may include an input/output cache. A cache controller keeps the cache filled with data or instructions that one or more masters are most likely to

need next. In this manner, information can be retrieved without delay.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a simplified diagram of a PCI bus architecture.

Figure 2 illustrates a simplified diagram of a request/grant architecture.

Figure 3 illustrates a flow chart of the priority scheme for a requesting master.

Figure 4 illustrates an example prioritization process including four requesting masters.

Figure 5 illustrates an exemplary request queue in accordance with the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a PCI bus architecture 100 including a PCI bus 106. Connected to PCI bus 106 are masters 101A, 101B, and 101C. A microprocessor 105 delegates arbitration of PCI bus 106 to an arbiter 103 (coupled to microprocessor 105 via a local bus 108) and is viewed by the system as another master in architecture 100. At this point, unless microprocessor 105 reassumes control, arbiter 103 assigns ownership of PCI bus 106 to requesting PCI masters.

Targets 102A and 102B, which can be accessed by any master, are connected to arbiter 103 via memory buses 107A and 107B, respectively. For purposes of this description, targets 102A and 102B include associated memories 104A and 104B, respectively.

In a PCI design, bus arbitration includes the following steps. A master asserts a request signal (REQ)

when it wants to take control of the bus. The arbiter sends a grant signal (GNT) to the master when permission is granted to take control of the bus. Referring to Figure 2, in a PCI bus, each master in request/grant architecture 200 has its own lines to request bus control and receive confirmation that control has been granted. Thus, each master has its own dedicated REQ# and GNT# lines, wherein # designates the master. When several masters request PCI bus ownership, each requesting master 101/103 asserts its respective request line. For example, master 101B makes its request using dedicated request line REQ#2. Arbiter 103 determines which master should get ownership of PCI bus 106 (Figure 1) and asserts the grant line associated with that master. For example, master 101B receives its grant signal from arbiter 103 via line GNT#2 (note that, for convenience, both the request/grant lines and the signals thereon are referred using the same designators).

In accordance with the present invention, dynamic information, such as the state of the target the master is accessing and the availability of the data the master device is requesting, is incorporated into the arbitration algorithm, thereby greatly enhancing system performance and maximizing system efficiency. Specifically, the arbiter logic in the present invention includes multiple levels of master priority.

In one embodiment, three levels of priority are provided: HIGH, MEDIUM, and LOW. As explained in further detail below, a master that has a request posted in the request queue of the target, but does not have data available in the target, has a LOW priority. Any master which does not have its request posted in the request queue of the target has a MEDIUM priority. Finally, a master

that has a request posted in the request queue of the target and data is available in the target has a HIGH priority.

Figure 3 illustrates a flow chart 300 that summarizes the priority scheme for each master in accordance with the present invention. After the prioritization process begins in step 301, the arbiter assigns a MEDIUM priority to the master in step 302. At this point, the master is inactive until the master asserts a request signal when it wants to take control of the bus in step 303. After the arbiter sends a provisional grant signal to the master in step 304, the arbiter determines whether data is available from the target associated with the request in step 305.

Assuming data is not available, then the arbiter assigns a LOW priority to the master in step 306. The master maintains this priority until the target data is available, as determined in step 307. At this point, the request posted by the master is modified in step 308 (i.e., a master ID is generated) and sent to the arbiter. After receiving the modified request, the arbiter changes the priority of the master to a HIGH priority in step 309.

Assuming the PCI bus is available and no other HIGH priority masters have earlier rights to the bus as determined in step 310, the arbiter sends a final grant signal to the master, thereby allowing the master to take control of the PCI bus. After data is transferred from the target in step 311, the arbiter returns to step 302 and changes the priority of the master back to a MEDIUM priority. Note that if target data is available in step 305, then the arbiter immediately modifies the request in step 308 and then proceeds through steps 309-311 as described above.

Figure 4 illustrates an example prioritization process in which masters 101A, 101B, 101C, and 105 (the microprocessor) each request bus control. At time t1, master 105 requests bus control. Therefore, the arbiter assigns master 105 a MEDIUM priority. At time t2, master 101A requests bus control at the same time that data becomes available to master 105. In accordance with the present invention, the arbiter assigns master 101A a MEDIUM priority and changes the priority of master 105 to a HIGH priority. Thus, master 105 is given control of the bus at time t2 until completion of its process at time t5.

In the interim, data is not available to master 101A at time t3. Therefore, at time t3, the arbiter reassigns master 101A a LOW priority. Also at time t3, master 101B requests bus control and is assigned a MEDIUM priority. At time t4, data is available to master 101B. Therefore, the arbiter reassigns master 101B a HIGH priority and allows master 101B to take control of the bus at time t5 (i.e. after the process associated with master 105 is complete). The process performed by master 101B ends at time t6.

Note that at time t5, master 101C requests bus control. However, because data is not available to master 101C at time t6, the arbiter reassigns master 101C a LOW priority. Thus, at this time, both masters 101A and 101C have a LOW priority. At time t7, data is available to master 101C. Therefore, the arbiter reassigns master 101C a HIGH priority and allows master 101C to take control of the bus at time t7.

At time t7, master 105 requests control of the bus and is therefore assigned a MEDIUM priority. At time t8, data becomes available to both masters 101A and 105. In accordance with the present invention, master 105

(previously having a MEDIUM priority) will take control of the bus before master 101A (previously having a LOW priority). Specifically, master 105 will take control at time t9 when the process performed by master 101C is completed. Master 101A will take control of the bus after the process performed by master 105 is complete.

In accordance with the present invention, each target 102 includes a memory interface to facilitate the prioritization process. **For example, r**eferring to Figure 2, target 102A includes an interface 201A to its associated memory 104A and target 102B includes an interface 201B to its associated memory 104B. The following example describes the functionality of memory interface 201A, wherein the functionality of memory interface 201B is identical to that of memory interface 201A. Suppose master 101A requests access to memory 104A. A request REQ#1 is sent to arbiter 103, which assigns master 101A a MEDIUM priority and forwards the request to target 102A. In one embodiment, request REQ#1 is sent to target 102A via memory bus 107a (Figure 1). However, in other embodiments, request REQ#1 is sent to target 102A via a dedicated line, just as the request/grant signals. Upon receipt of REQ#1, target 102A proceeds to capture the information needed to complete the access of memory 104A.

The captured information is stored in a request queue in the interface 201A of target 102A. Figure 5 illustrates an exemplary request queue 500 including request information 501, a plurality of master identifications (IDs) 502, and data 503. In one embodiment, target 102A generates a master ID 502A using request REQ#1 (i.e., request information 501A) after data 503A is available and stored in request queue 500. Master ID 502A, shown as

modified request MREQ#1 in Figure 2, is then provided to arbiter 103. In one embodiment, modified request MREQ#1 is sent to arbiter 103 via memory bus 107a (Figure 1). However, in other embodiments, modified request MREQ#1 is sent to arbiter 103 via a dedicated line, just as the request/grant signals.

After arbiter 103 receives modified request MREQ#1, arbiter 103 changes the priority of master 101A to HIGH and, and assuming PCI bus 106 (Figure 1) is available and no other masters have an earlier HIGH priority, sends a final grant signal GNT#1 to master 101A, thereby allowing master 101A to take control of PCI bus 106. Note that if PCI bus 106 is currently being controlled by another master or if other masters have an earlier HIGH priority, then arbiter 103 allows master 101A to take control of PCI bus 106 at the next earliest time period after the process performed by the last controlling master is complete.

As described in reference to Figure 4, if data is not available in the next time period (time t3) following request REQ#1 (time t2), then arbiter 103 changes the priority of master 101A to LOW. In this manner, master 101A need not repeat, and indeed in a preferred embodiment is prevented from repeating, REQ#1 if target 102A is not ready to supply the data. As described previously, during the period that master 101A is assigned a LOW priority (in Figure 4, from time t3 to time t8), arbiter 103 allocates the ownership of PCI bus 106 to another PCI master requesting the bus and having data available (such as master 101B at time t5 and master 101C at time t7).

In accordance with one embodiment of the present invention, target 102A can post different requests into the request queue of interface 201A. In this manner, multiple

masters can access the memory of the same target (In fact, because of the sequential servicing of the present invention, multiple masters can even access the same data in a memory.) Elaborating upon the example including master 101A above, assume that master 101B also sends a request REQ#2 to arbiter 103 to access memory 104A of target 102A. In this case, target 102A posts REQ#2 (i.e., request information 501B) from master 101B in request queue 500 (Figure 4). Note that although only two requests can be processed by request queue 500, other embodiments can include any number of requests with corresponding master Ids and data. Assume further that the data requested by master 101B is available before the data requested by master 101A. In the present invention, a modified request signal MREQ#2 (i.e., Master ID 502B of Figure 5)(neither shown in Figure 2) is provided to arbiter 103 before modified request MREQ#1 is provided. Thus, the priority of master 101B changes to HIGH before that of master 101A.

To further increase the efficiency of the present invention, the request queue may include an input/output cache. The cache interposes a block of fast memory between a memory 104 and arbiter 103. A cache controller keeps the cache filled with data or instructions that one or more masters are most likely to need next. In this manner, information can be retrieved without delay (see Figure 4, for example, in which master 105 posts a request at time t1 and retrieves information at time t2). For most system applications, a cache size of approximately 256 KB is satisfactory.

In one embodiment, a PCI bridge implements the above-described priority scheme of the present invention. As known by those skilled in the art, a bridge is a system

building block used to transport data between various buses. A bridge can connect different types of buses. Thus, referring to Figure 1, a bridge can interconnect PCI bus 106, memory buses 107, and local bus 108. A bridge may be an ASIC device, or may be part of a chipset in the system. In a preferred embodiment, the PCI bridge includes a chipset that integrates targets 102 and arbiter 103.

In summary, the present invention includes a priority scheme implemented by the arbiter in conjunction with the memory interfaces in the target devices. The priority of various requesting masters changes dynamically based on data availability and request state. Using the above features, the present invention maximizes utilization of the PCI bandwidth and throughput.

The specific embodiments of the present invention are presented for purposes of description and illustration only. These embodiments are not intended to be exhaustive or to limit the invention in any way. Those skilled in the art will recognize modifications and variations to the present invention. The present invention is only defined by the appended claims.

CLAIMS

I claim:

1. A method of assigning ownership of a peripheral component interconnect (PCI) bus, the method including:

identifying a target requested by a master;

determining if data associated with the target is available; and

assigning a priority to the master based on availability of the data.

2. The method of Claim 1, wherein after the master requests a target, the master is assigned a MEDIUM priority.

3. The method of Claim 1, wherein if the data is not available, then the master is assigned a LOW priority.

4. The method of Claim 1, wherein if the data is available, then the master is assigned a HIGH priority.

5. The method of Claim 1, wherein the target uses delayed transactions to complete a read access.

6. The method of Claim 5, wherein the target integrates a buffer management scheme.

7. The method of Claim 6, wherein the buffer management scheme includes an input/output cache.

8. The method of Claim 1, wherein identifying a target includes sending a request signal from the master to an arbiter.

9. The method of Claim 8, wherein assigning a priority includes sending a modified request signal to the arbiter.

PERIPHERAL COMPONENT INTERCONNECT ARBITER IMPLEMENTATION WITH DYNAMIC PRIORITY SCHEME

Sujith Arramreddy
Ragu Appanagari

ABSTRACT OF THE INVENTION

A dynamic priority scheme is provided that uses information including the status of the target and data availability in deciding which PCI master should be assigned ownership of the bus. The target uses delayed transactions to complete a read access targeted to it. The target also integrates a buffer management scheme, in one embodiment an input/output cache, for buffer management. The present invention optimizes the performance and utilization of the PCI bus.

Figure 1

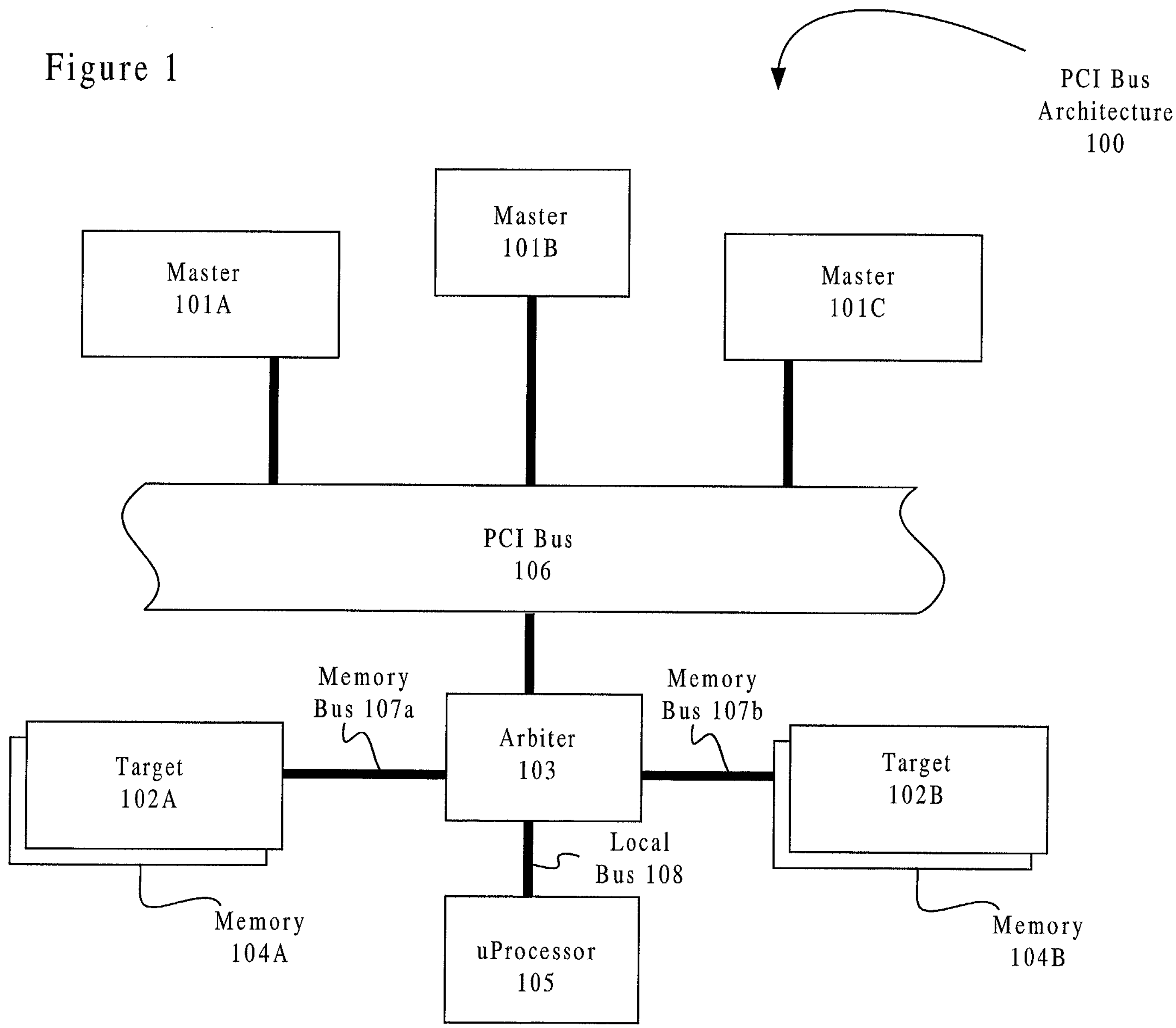
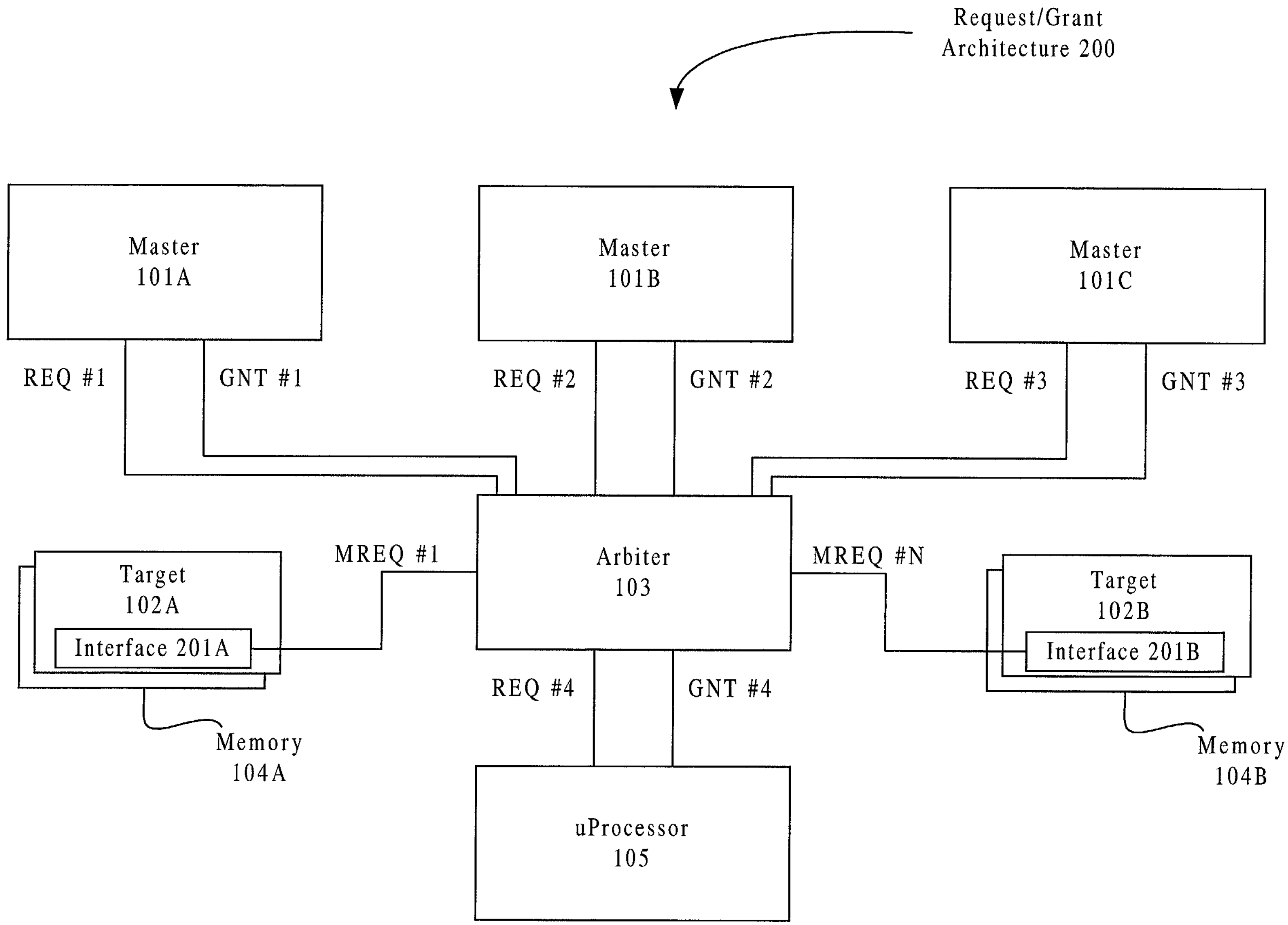

PCI Bus
Architecture
100
Master
101A
Master
101B
Master
101C
PCI Bus
106
Memory
Bus 107a
Memory
Bus 107b
Arbiter
103
Target
102A
Target
102B
Local
Bus 108
Memory
104A
Memory
104B
uProcessor
105

Figure 2



Request/Grant
Architecture 200
Master
101A
Master
101B
Master
101C
REQ #1
GNT #1
REQ #2
GNT #2
REQ #3
GNT #3
Target
102A
Interface 201A
MREQ #1
Arbiter
103
MREQ #N
Target
102B
Interface 201B
REQ #4
GNT #4
Memory
104A
Memory
104B
uProcessor
105

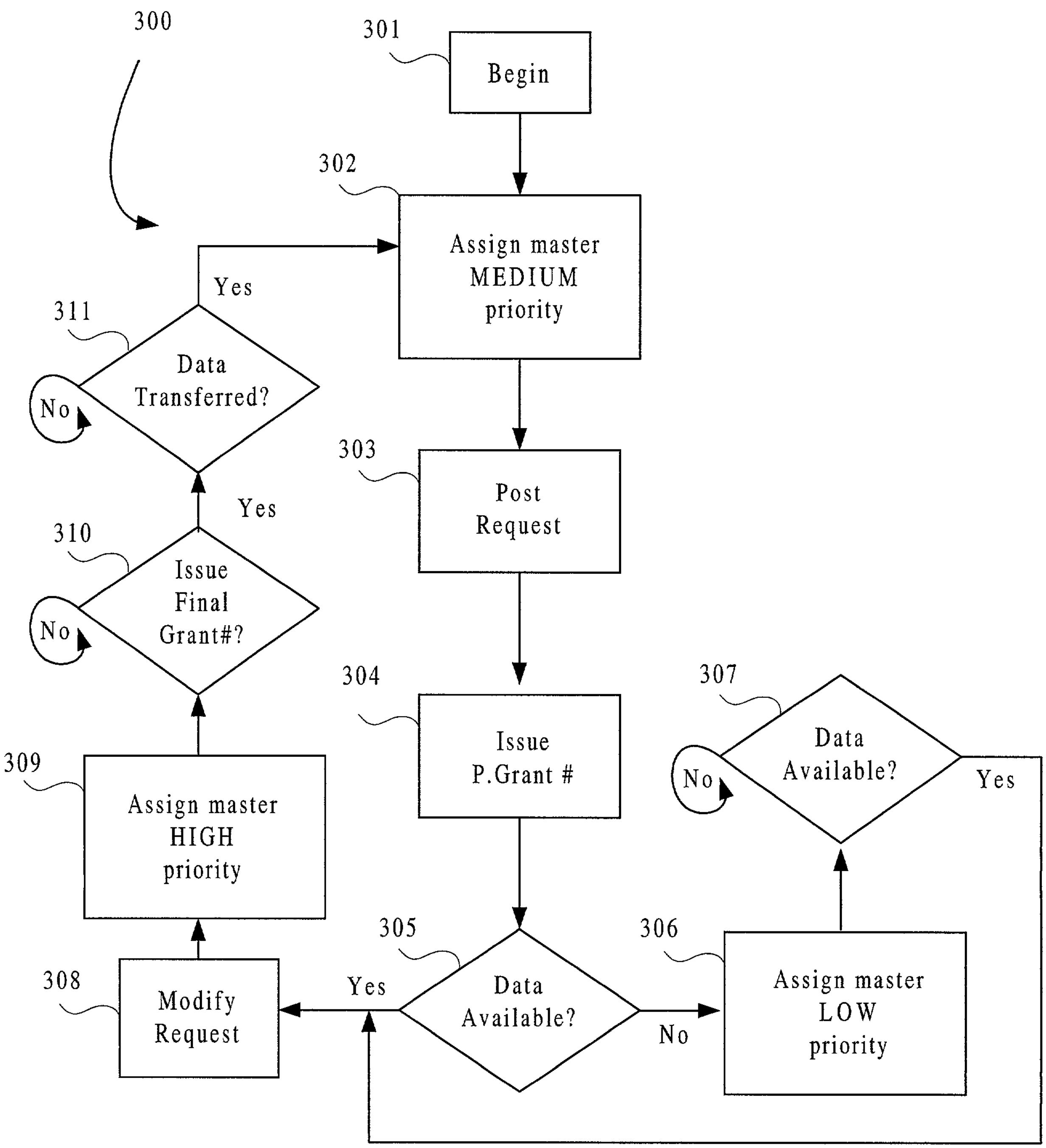

300
301
Begin
302
Assign master
MEDIUM
priority
303
Post
Request
304
Issue
P.Grant #
305
Data
Available?
Yes
No
306
Assign master
LOW
priority
307
Data
Available?
No
Yes
308
Modify
Request
309
Assign master
HIGH
priority
310
Issue
Final
Grant#?
No
Yes
311
Data
Transferred?
No
Yes

Figure 3

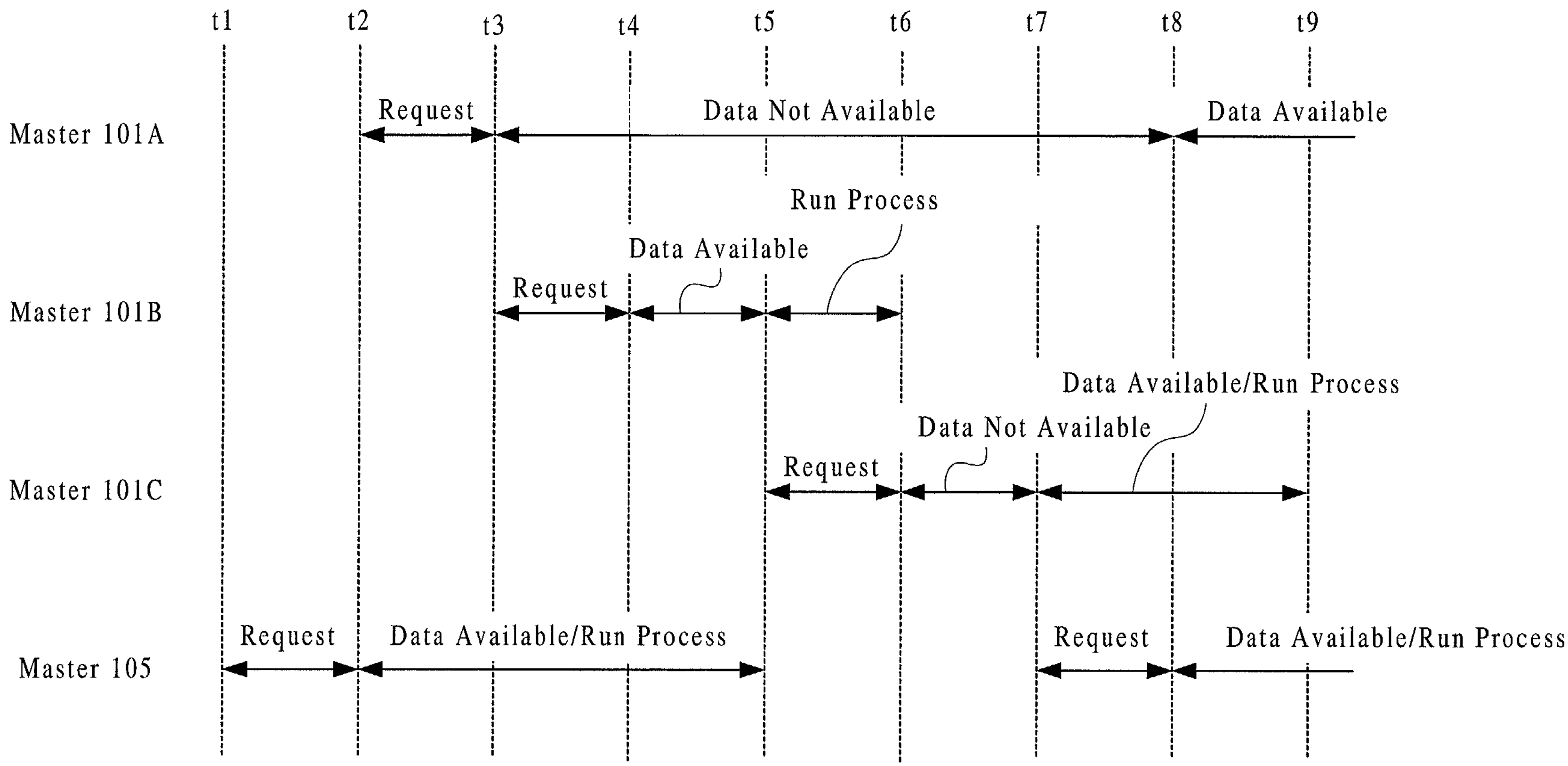

t1
t2
t3
t4
t5
t6
t7
t8
t9
Master 101A
Request
Data Not Available
Data Available
Master 101B
Request
Data Available
Run Process
Master 101C
Request
Data Not Available
Data Available/Run Process
Master 105
Request
Data Available/Run Process
Request
Data Available/Run Process

Figure 4

500

Request Information 501A	Master ID 502A	Data 503A
Request Information 501B	Master ID 502B	Data 503B

Figure 5